NASP–9214–06
NAS Enroute Stage A
Contract FA65WA–1395

19 NOV 1973

9020 DOCUMENT
CONTROL SECTION

NAS OPERATIONAL SUPPORT SYSTEM

USER'S MANUAL

IBM 9020 Data Processing System Basic Assembly Language (BALASM)

Model A3d2.0

COPY No. 04

15 August 1973

The Basic Assembly Language is a symbolic programming language that provides a convenient way to make full use of the capabilities of the IBM 9020 Data Processing System. The assembly language provides flexibility in program and machine usage and assists the programmer in checking, standardizing, and documenting programs. This revision conforms with the C42 release of the BAL Assembler.

This document replaces NASP–9214–05 and is compatible with the NOSS tapes which support the NAS Model A3d2.0 tape release.

NAS Programming
IBM Federal Systems Division
NAFEC, Atlantic City, New Jersey

SECTION 3. MACHINE INSTRUCTION STATEMENT

All machine instructions may be represented symbolically as assembly language statements. The symbolic format of each statement varies according to the actual machine-instruction format, of which there are five: RR, RX, RS, SI, and SS. Within each basic format, further variations are possible.

The symbolic format of a machine instruction parallels, but does not duplicate, its actual format. A mnemonic operation code is written in the operation field, and one or more operands are written in the operand field. Comments may be appended to a machine-instruction statement as previously explained in Section 1.

Any machine-instruction statement may be named by a symbol, which other assembly statements can use as an operand. The value attribute of the symbol is the address of the leftmost byte assigned to the assembled instruction. The length attribute of the symbol depends on the basic instruction format, as follows:

Basic Format	Implied Length in Bytes
RR	2
RX	4
RS	4
SI	4
SS	6

INSTRUCTION ALIGNMENT AND CHECKING

All machine instructions are aligned automatically by the assembly program on halfword boundaries. Bytes that are skipped because of alignment are set to zero. All expressions that specify storage addresses are checked to ensure that they refer to the proper boundaries (halfword, fullword, or doubleword) for the instructions in which they are used. If a base register and displacement are explicit, the displacement is checked. For statements with an implied base register and displacement, the effective address is checked. Register numbers are also checked to ensure that they specify the proper registers, as follows:

1. Floating-point instructions must specify floating-point registers 0, 2, 4, or 6.

2. Double shift, fullword multiply, and fullword divide instructions must specify an even-numbered general register.

OPERAND FORMAT

Table 3–1 shows the symbolic operand formats for each variation of the basic machine-instruction formats. The table also shows the size, in bits, of each machine-instruction field (see the number at the top of each field) and it designates the instructions that use each operand format. All symbolic operand formats are indicated in the table as a series of codes (e.g., R1, S2, D2, and L). These codes are explained in the notes for Table 3–1. The notes provide other information pertinent to the use of symbolic operands and should be considered an integral part of the table.

Note: More than one operand format is shown for each variation of each machine-instruction format, except the RR type. The selection of an operand pattern depends on what is to be specified and whether it is to be specified explicitly or implicitly. This is discussed in the subsections on "Effective Addresses – – – Explicit and Implied" and "Lengths – – – Explicit and Implied."

Operand Fields and Subfields

Table 3–1 shows that some symbolic operands are written as a single field and other operands are written as a field followed by one or two subfields. For example, effective addresses consist of the contents of a base register and a displacement. A symbolic operand that specifies a base and displacement is written as a displacement field followed by a base register subfield. In the RX format, both an index register subfield and a base register subfield are written; in the SS format, both a length subfield and a base register subfield are written.

A comma must be written to separate operands. Parentheses must be written to enclose a subfield or subfields, and a comma must be written to separate two subfields within parentheses. When parentheses are used to enclose one subfield, and the subfield is omitted, the parentheses must be omitted. In the case of two subfields that are separated by a comma and enclosed by parentheses, the following rules apply:

1. If both subfields are omitted, the separating comma and the parentheses must also be omitted.

2. If the first subfield in the sequence can be omitted, the comma that separates it from the second subfield is written. The parentheses must also be written.

3. If the second subfield in the sequence is omitted, the comma that separates it from the first subfield must be omitted. The parentheses must be written.

All fields and subfields in a symbolic operand may be represented either by absolute or relocatable expressions, depending on what the field requires. (An expression consists of one term or a series of arithmetically combined terms.)

Note: Blanks may not appear in an operand unless provided by a character self-defining value or a character literal. Thus, blanks may not intervene between fields and the comma separators, between parentheses and fields, etc.

Table 3–1. Machine Instruction Format

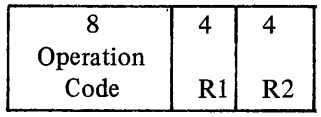
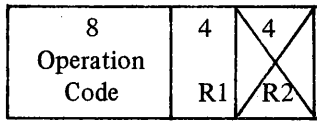
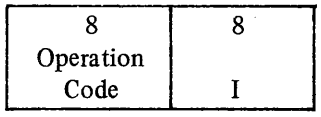
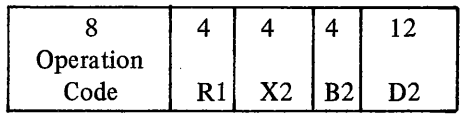
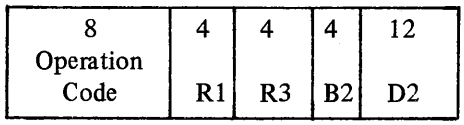
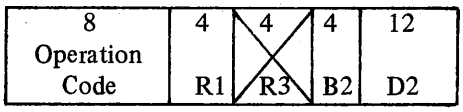
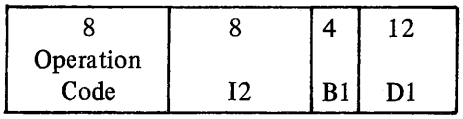
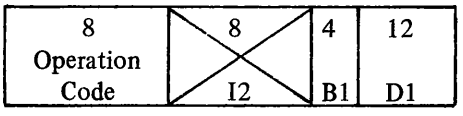
	Basic Machine Format	Operand Field Format	Applicable Instructions
RR		R1,R2	All RR instructions except DLY, LI, SPM, and SVC
		R1	SPM, LI
		I	SVC, DLY
RX		R1, D2(X2,B2) R1, S2(X2)	All RX instructions
RS		R1, R3, D2(B2) R1, R3, S2	BXH, BXLE, LM, and STM
		R1, D2(B2) R1, S2 R1, I2	All shift instructions
SI		D1(B1), I2 S1,I2	All SI instructions except LPSW, SSM, SPCI, HIO, SIO, TIO, TCH, LPSB and SPSB
		D1, (B1) S1	LPSW, SSM, HIO, SIO, TIO, TCH, SPCI, LPSB, SPSB

Table 3–1. Machine Instruction Format (Continued)

Basic Machine Format								Operand Field Format	Applicable Instructions
	8 Operation Code	4 L1	4 L2	4 B1	12 D1	4 B2	12 D2	D1(L1, B1), D2(L2, B2) S1(L), S2(L2)	PACK, UNPK, MVO, AP, CP, DP, MP, SP, ZAP
SS	8 Operation Code	8 L	4 B1	12 D1	4 B2	12 D2		D1(L, B1), D2(B2) S1(L), S2	NC, OC, XC, CLC, MVC, MVN, MVZ, TR, TRT, ED, EDMK

Notes for Table 3–1

1. *R1, R2, and R3 are absolute expressions that specify general or floating-point registers. The general register numbers are 0 through 15; floating-point register numbers are 0, 2, 4, and 6.*
2. *D1 and D2 are absolute expressions that specify displacements. A value of 0–4095 may be specified. See the subsequent discussion on effective addresses and Table 3–2.*
3. *B1 and B2 are absolute expressions that specify base registers. Register numbers are 0–15. See the subsequent discussion on effective addresses and Table 3–2.*
4. *X2 is an absolute expression that specifies an index register. Register numbers are 0–15. If B2 is specified, X2 should not be omitted; and when indexing is not desired, X2 must be specified as 0. See Table 3–2.*
5. *L, L1, and L2 are absolute expressions that specify field lengths. An L expression can specify a value of 1–256. L1 and L2 expressions can specify a value of 1–16. In all cases, the assembled value will be one less than the specified value. See the subsequent discussion on explicit and implied lengths and Table 3–3. An L, L1, or L2 field of 0 is equivalent to 1.*
6. *I and I2 are absolute expressions that provide immediate data. The value of the expression may be 0–255.*
7. *S1 and S2 are absolute or relocatable expressions that specify an effective address. See the subsequent discussion on effective addresses and Table 3–2.*
8. *RR, RS, and SI instruction fields that are crossed out in the machine formats are not examined during instruction execution. The fields are not written in the symbolic operand, but are assembled as binary zeros.*

Effective Addresses – Explicit and Implied

An effective address is composed of a displacement plus the contents of a base register. (In the case of RX instructions, the contents of an index register are also used to derive the effective address.) The programmer writes an explicit effective address by specifying the displacement and the base register number. He may also write an implied effective address by specifying an absolute or relocatable address. The assembly program has the facility to select a base register and compute a displacement, thereby generating an explicit address from an implied address, provided that it has been informed (1) what base registers are available to it and (2) what each register contains. The programmer conveys this information to the assembly program through the USING and DROP assembly instructions which are described in Section 5 under "Base Register Instructions." Additional addressing considerations pertain when programs have more than one control section. These considerations also are discussed in "Base Register Instructions."

Table 3–1 shows two types of addressing formats for RX, RS, SI, and SS instructions. In each case, the first type shows the method of specifying an effective address explicitly, as a base register and displacement. The second type indicates how to specify an effective address implicitly, as an expression. For example, the load multiple instruction (RS format) may have either of the following symbolic operands:

R1, R3, D2(B2)	**Explicit address**
R1, R3, S2	**Implicit address**

Whereas D2 and B2 must be represented by absolute expressions, S2 may be represented either by a relocatable or an absolute expression.

In order to use implicit addresses, the following rules must be observed:

1. The base register assembly instructions (USING and DROP) must be used as explained in Section 4.

2. An explicit base register must not accompany the implicit address.

For example, assume that FIELD is a relocatable symbol, which has been assigned a value of 7400. Assume also that the assembly program has been notified (by a USING instruction) that general register 12 currently contains a relocatable value of 4096 and is available as a base register. The following example shows a machine-instruction statement as it would be written in assembly language and as it would be assembled. Note that the value of D2 is the difference between 7400 and 4096 and that X2 is assembled as zero, because it was omitted. The assembled instruction is presented in decimal (op code is in hexadecimal).

Assembly language statement:

ST 4,FIELD

Assembled instruction:

Op Code	R1	X2	B2	D2
50	4	0	12	3304

To summarize, an effective address may be specified explicitly as a base register and displacement (and index register for RX instructions) by the formats shown in the first column of Table 3–2. The address may be specified implicitly by the formats shown in the second column. Table 3–2 is an elaboration of the operand information summarized in notes 2, 3, 4, 7, and 8 for Table 3–1. Note that the two storage addresses required by the SS instructions are presented separately; an implicit address may be used for one while an explicit address is used for the other.

Table 3–2. Details of Address Specification

Type	Explicit Effective Address	Implicit Effective Address
RX	D2(X2,B2)	S2(X2)
	D2(0,B2)*	S2
RS	D2(B2)	S2
SI	D1(B1)	S1
SS	D1(L1,B1)	S1(L1)
	D1(L,B1)	S1(L)
	D2(L2,B2)	S2(L2)
	D1(B1)	S1
	D2(B2)	S2

*A zero must be supplied when it is desired to omit an index register specification in an RX explicit address, or the base register will be interpreted as the index register. Note that this will not cause the instruction to execute differently.

Lengths – Explicit and Implied

The length field in SS instructions can be explicit or implied. To imply a length, the programmer omits a length field from the operand. The omission indicates that the length field is either of the following:

1. The length attribute of the expression specifying the displacement, if an explicit base and displacement have been written.

2. The length attribute of the expression specifying the effective address, if the base and displacement have been implied.

In either case, the length attribute for a multiterm expression is the length of the leftmost term in the expression.

By contrast, an explicit length is written by the programmer in the operand as an absolute expression. The explicit length overrides any implied length.

Whether the length is explicit or implied, it is always an effective length. The value inserted into the length field of the assembled instruction is one less than the effective length in the machine-instruction statement.

Although length fields are usually in terms of bytes, the MVW instruction is an exception to this, i.e., its length field specifies words.

To summarize, the length required in an SS instruction may be specified explicitly by the formats shown in the first column of Table 3–3 or may be implied by the formats shown in the second column. This table is an elaboration of the operand information summarized in note 5 for Table 3–1. Note that the two lengths required in one of the SS instruction formats are presented separately; an implied length may be used for one while an explicit length is used for the other.

Table 3–3. Details of Length Specification in SS Instruction

Explicit Length	Implicit Length
D1(L1,B1)	D1(,B1) (Implicit Length of 1)
S1(L1)	S1
D1(L,B1)	D1(,B1) (Implicit Length of 1)
S1(L)	S1
D2(L2,B2)	D2(,B2)
S2(L2)	S2

MACHINE INSTRUCTION MNEMONIC CODES

This subsection contains an alphabetical listing of the mnemonic operation codes of all the machine instructions that can be represented in the assembly language. The column headings in Table 3–4 and the information each column provides are:

Mnemonic Code: This column gives the mnemonic operation code for the machine instruction.

Instruction: This column contains the name of the instruction associated with the mnemonic.

Operation Code: This column contains the hexadecimal equivalent of the actual machine operation code.

Basic Machine Format: This column gives the basic machine format of the instruction: RR, RX, RS, SI, or SS.

Operand Field Format: This column shows the symbolic format of the operand field for the particular mnemonic.

The mnemonic operation codes are designed to be easily remembered codes that indicate the functions of the instructions. The normal format of the code is shown below; the items in brackets are not necessarily present in all codes:

Verb [Modifier] [Data Type] [Machine Format]

The verb, which is usually one or two characters, specifies the function. For example, A represents Add. and MV represents Move. The function may be further defined by a modifier. For example, the modifier L indicates a logical function, as an AL for Add Logical.

Mnemonic codes for functions involving data usually indicate the data types by letters that correspond to those for the data types in the DC assembly instruction (see Section 4). Furthermore, letters U and W have been added to indicate short and long, unnormalized floating-point operations, respectively. For example, AE indicates Add Normalized Short, whereas AU indicates Add Unnormalized Short. Where applicable, fullword fixed-point data is implied if the data type is omitted.

The letters R and I are added to the codes to indicate, respectively, RR and SI machine-instruction formats. Thus, AER indicates Add Normalized Short in the RR format. Functions involving character and decimal data types imply the SS format.

Machine Instruction Examples

The examples that follow are grouped according to machine-instruction format. They illustrate the various symbolic operand formats, which are summarized in Table 3–1 and detailed in Tables 3–2 and 3–3. All symbols employed in the examples must be assumed to be defined elsewhere in the same assembly. All symbols that specify register numbers and lengths must be assumed to be equated to absolute values elsewhere.

Implied addressing, control section addressing, and the function of the USING assembly instruction are not considered here. For discussion of these considerations and for examples of coding sequences that illustrate them, refer to Section 4, "Program Sectioning and Linking Instructions" and "Base Register Instructions."

RR FORMAT

Name	Operation	Operand
ALPHA1	LR	1,2
ALPHA2	LR	REG1,REG2
BETA	SPM	15
GAMMA1	SVC	250
GAMMA2	SVC	TEN

The operands of ALPHA1, BETA, and GAMMA1 are decimal self-defining values, which are categorized as absolute expressions. The operands of ALPHA2 and GAMMA2 are symbols that are equated to absolute values elsewhere.

Table 3–4. Machine Instruction Mnemonic Codes

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
A	Add	5A	RX	R1,D2(X2,B2)
AD	Add Normalized, Long	6A	RX	R1,D2(X2,B2)
ADR	Add Normalized, Long	2A	RR	R1,R2
AE	Add Normalized, Short	7A	RX	R1,D2(X2,B2)
AER	Add Normalized, Short	3A	RR	R1,R2
AH	Add Halfword	4A	RX	R1,D2(X2,B2)
AL	Add Logical	5E	RX	R1,D2(X2,B2)
ALR	Add Logical	1E	RR	R1,R2
AP	Add Decimal	FA	SS	D1(L1,B1),D2(L2,B2)
AR	Add	1A	RR	R1,R2
AU	Add Unnormalized, Short	7E	RX	R1,D2(X2,B2)
AUR	Add Unnormalized, Short	3E	RR	R1,R2
AW	Add Unnormalized, Long	6E	RX	R1,D2(X2,B2)
AWR	Add Unnormalized, Long	2E	RR	R1,R2
BAL	Branch and Link	45	RX	R1,D2(X2,B2)
BALR	Branch and Link	05	RR	R1,R2
BC	Branch on Condition	47	RX	R1,D2(X2,B2)
BCR	Branch on Condition	07	RR	R1,R2
BCT	Branch on Count	46	RX	R1,D2(X2,B2)
BCTR	Branch on Count	06	RR	R1,R2
BXH	Branch on Index High	86	RS	R1,R3,D2(B2)
BXLE	Branch on Index Low or Equal	87	RS	R1,R3,D2(B2)
C	Compare Algebraic	59	RX	R1,D2(X2,B2)
CD	Compare, Long	69	RX	R1,D2(X2,B2)
CDR	Compare, Long	29	RR	R1,R2
CE	Compare, Short	79	RX	R1,D2(X2,B2)
CER	Compare, Short	39	RR	R1,R2
CH	Compare Halfword	49	RX	R1,D2(X2,B2)
CL	Compare Logical	55	RX	R1,D2(X2,B2)
CLC	Compare Logical	D5	SS	D1(L,B1),D2(B2)
CLI	Compare Logical Immediate	95	SI	D1(B1),I2
CLR	Compare Logical	15	RR	R1,R2
CP	Compare Decimal	F9	SS	D1(L1,B1),D2(L2,B2)
CR	Compare Algebraic	19	RR	R1,R2
CSS	Convert and Sort Symbols	02	RR	R1,R2
CVB	Convert to Binary	4F	RX	R1,D2(X2,B2)
CVD	Convert to Decimal	4E	RX	R1,D2(X2,B2)
CVWL	Convert Weather Lines	03	RR	R1,R2
D	Divide	5D	RX	R1,D2(X2,B2)
DD	Divide, Long	6D	RX	R1,D2(X2,B2)
DDR	Divide, Long	2D	RR	R1,R2
DE	Divide, Short	7D	RX	R1,D2(X2,B2)
DER	Divide, Short	3D	RR	R1,R2
DIAG	Diagnose	83	SI	DI(B1),I2
DLY	Delay	0B	RR	I
DP	Divide Decimal	FD	SS	D1(L1,B1),D2(L2,B2)
DR	Divide	1D	RR	R1,R2
ED	Edit	DE	SS	D1(L,B1),D2(B2)

Table 3–4. Machine Instruction Mnemonic Codes (Continued)

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
EDMK	Edit and Mark	DF	SS	D1(L1,B1),D2(B2)
EX	Execute	44	RX	R1,D2(X2,B2)
HDR	Halve, Long	24	RR	R1,R2
HER	Halve, Short	34	RR	R1,R2
HIO	Halt Input/Output	9E	SI	D1(B1)
IATR	Insert Address Translator	0E	RR	R1,R2
IC	Insert Character	43	RX	R1,D2(X2,B2)
ISK	Insert Storage Key	09	RR	R1,R2
L	Load	58	RX	R1,D2(X2,B2)
LA	Load Address	41	RX	R1,D2(X2,B2)
LC	Load Chain	52	RX	R1,D2(X2,B2)
LCDR	Load Complement, Long	23	RR	R1,R2
LCER	Load Complement, Short	33	RR	R1,R2
LCR	Load Complement	13	RR	R1,R2
LD	Load, Long	68	RX	R1,D2(X2,B2)
LDA	Load Data Address	99	RS	R1,D2(B2)
LDR	Load, Long	28	RR	R1,R2
LE	Load, Short	78	RX	R1,D2(X2,B2)
LER	Load, Short	38	RR	R1,R2
LH	Load Halfword	48	RX	R1,D2(X2,B2)
LI	Load Identity	0C	RR	R1
LM	Load Multiple	98	RS	R1,R3,D2(B2)
LNDR	Load Negative, Long	21	RR	R1,R2
LNER	Load Negative, Short	31	RR	R1,R2
LNR	Load Negative	11	RR	R1,R2
LPDR	Load Positive, Long	20	RR	R1,R2
LPER	Load Positive, Short	30	RR	R1,R2
LPR	Load Positive	10	RR	R1,R2
LPSB	Load Preferential Storage Base	A1	SI	D1(B1)
LPSW	Load PSW	82	SI	D1(B1)
LR	Load	18	RR	R1,R2
LTDR	Load and Test, Long	22	RR	R1,R2
LTER	Load and Test, Short	32	RR	R1,R2
LTR	Load and Test	12	RR	R1,R2
M	Multiply	5C	RX	R1,D2(X2,B2)
MD	Multiply, Long	6C	RX	R1,D2(X2,B2)
MDR	Multiply, Long	2C	RR	R1,R2
ME	Multiply, Short	7C	RX	R1,D2(X2,B2)
MER	Multiply, Short	3C	RR	R1,R2
MH	Multiply Halfword	4C	RX	R1,D2(X2,B2)
MP	Multiply Decimal	FC	SS	D1(L1,B1),D2(L2,B2)
MR	Multiply	1C	RR	R1,R2
MVC	Move Characters	D2	SS	D1(L,B1),D2(B2)
MVI	Move Immediate	92	SI	D1(B1),I2
MVN	Move Numerics	D1	SS	D1(L,B1),D2(B2)
MVO	Move with Offset	F1	SS	D1(L1,B1),D2(L2,B2)
MVW	Move Word	D8	SS	D1,(L,B1),D2(B2)

Table 3–4. Machine Instruction Mnemonic Codes (Continued)

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
MVZ	Move Zones	D3	SS	D1(L,B1),D2(B2)
N	AND Logical	54	RX	R1,D2(X2,B2)
NC	AND Logical	D4	SS	D1(L,B1),D2(B2)
NI	AND Logical Immediate	94	SI	D1(B1),I2
NR	AND Logical	14	RR	R1,R2
O	OR Logical	56	RX	R1,D2(X2,B2)
OC	OR Logical	D6	SS	D1(L,B1),D2(B2)
OI	OR Logical Immediate	96	SI	D1(L,B1),I2
OR	OR Logical	16	RR	R1,R2
PACK	Pack	F2	SS	D1(L1,B1),D2(L2,B2)
RDD	Read Direct	85	SI	D1(B1),I2
RPSB	Repack Symbols	0F	RR	R1,R2
S	Subtract	5B	RX	R1,D2(X2,B2)
SATR	Set Address Translator	0D	RR	R1,R2
SCON	Set Configuration	01	RR	R1,R2
SD	Subtract Normalized, Long	6B	RX	R1,D2(X2,B2)
SDR	Subtract Normalized, Long	2B	RR	R1,R2
SE	Subtract Normalized, Short	7B	RX	R1,D2(X2,B2)
SER	Subtract Normalized, Short	3B	RR	R1,R2
SH	Subtract Halfword	4B	RX	R1,D2(X2,B2)
SIO	Start Input/Output	9C	SI	D1(B1)
SIOP	Start I/O Processor	9A	SI	D1(B1),I2
SL	Subtract Logical	5F	RX	R1,D2(X2,B2)
SLA	Shift Left Short Algebraic	8B	RS	R1,D2(B2)
SLDA	Shift Left Long Algebraic	8F	RS	R1,D2(B2)
SLDL	Shift Left Double Logical	8D	RS	R1,D2(B2)
SLL	Shift Left Short Logical	89	RS	R1,D2(B2)
SLR	Subtract Logical	1F	RR	R1,R2
SP	Subtract Decimal	FB	SS	D1(L1,B1),D2(L2,B2)
SPCI	Set PCI	9B	SI	D1(B1)
SPM	Set Program Mask	04	RR	R1
SPSB	Store Preferential Storage Base	A0	SI	D1(B1)
SR	Subtract	1B	RR	R1,R2
SRA	Shift Right Short Algebraic	8A	RS	R1,D2(B2)
SRDA	Shift Right Long Algebraic	8E	RS	R1,D2(B2)
SRDL	Shift Right Long Logical	8C	RS	R1,D2(B2)
SRL	Shift Right Short Logical	88	RS	R1,D2,(B2)
SSK	Set Storage Key	08	RR	R1,R2
SSM	Set System Mask	80	SI	D1(B1)
ST	Store	50	RX	R1,D2(X2,B2)
STC	Store Character	42	RX	R1,D2(X2,B2)
STD	Store Long	60	RX	R1,D2(X2,B2)
STE	Store Short	70	RX	R1,D2(X2,B2)
STH	Store Halfword	40	RX	R1,D2(X2,B2)
STM	Store Multiple	90	RS	R1,R3,D2(B2)
SU	Subtract Unnormalized, Short	7F	RX	R1,D2(X2,B2)
SUR	Subtract Unnormalized, Short	3F	RR	R1,R2
SVC	Supervisor Call	0A	RR	I

Table 3–4. Machine Instruction Mnemonic Codes (Continued)

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
SW	Subtract Unnormalized, Long	6F	RX	R1,D2(X2,B2)
SWR	Subtract Unnormalized, Long	2F	RR	R1,R2
TCH	Test Channel	9F	SI	D1(B1)
TIO	Test Input/Output	9D	SI	D1(B1)
TM	Test Under Mask	91	SI	D1(B1),I2
TR	Translate	DC	SS	D1(L,B1),D2(B2)
TRT	Translate and Test	DD	SS	D1(L,B1),D2(B2)
TS	Test and Set	93	SI	D1(B1)
UNPK	Unpack	F3	SS	D1(L1,B1),D2(L2,B2)
WRD	Write Direct	84	SI	D1(B1),I2
X	Exclusive OR	57	RX	R1,D2(X2,B2)
XC	Exclusive OR	D7	SS	D1(L,B1),D2(B2)
XI	Exclusive OR, Immediate	97	SI	D1(B1),I2
XR	Exclusive OR	17	RR	R1,R2
ZAP	Zero and Add Decimal	F8	SS	D1(L1,B1),D2(L2,B2)

RX FORMAT

Name	Operation	Operand
ALPHA1	L	1, 39(4,10)
ALPHA2	L	REG1, 39(4,TEN)
BETA1	L	2, ZETA(4)
BETA2	L	REG2,ZETA(REG4)
GAMMA1	L	2,ZETA
GAMMA2	L	REG2, ZETA
GAMMA3	L	2, =F '1000'

Both ALPHA instructions specify explicit addresses; REG1 and TEN are absolute symbols. Both BETA instructions specify implicit addresses, and both use index registers. Indexing is omitted from the GAMMA instructions. GAMMA1 and GAMMA2 specify implicit addresses. The second operand of GAMMA3 is a literal.

RS FORMAT

Name	Operation	Operand
ALPHA1	BXH	1,2,20(14)
ALPHA2	BXH	REG1,REG2,20(REGD)
ALPHA3	BXH	REG1,REG2,ZETA
BETA1	SLL	1,20(9)
BETA2	SLL	REG1,20
BETA3	SLL	REG1,ZETA

Whereas ALPHA1 and ALPHA2 specify explicit addresses, ALPHA3 specifies an implicit address. Similarly, the BETA instructions illustrate both explicit and implicit addresses.

SI FORMAT

Name	Operation	Operand
ALPHA1	CLI	40(9),X'40'
ALPHA2	CLI	40(REG9),TEN
BETA1	CLI	ZETA, TEN
BETA2	CLI	ZETA,C 'A'
GAMMA1	SIO	40(9)
GAMMA2	SIO	0(9)
GAMMA3	SIO	40(0)
GAMMA4	SIO	ZETA

The ALPHA instructions and GAMMA1–GAMMA3 specify explicit addresses, whereas the BETA instructions and GAMMA4 specify implicit addresses. GAMMA2 specifies a displacement of zero. GAMMA3 does not specify a base register.

SS FORMAT

Name	Operation	Operand
ALPHA1	AP	40(9,8),30(6,7)
ALPHA2	AP	40(NINE,REG8), 30(REG6,7)
ALPHA3	AP	FIELD2,FIELD1
ALPHA4	AP	FIELD1(6),FIELD2(9)
BETA	AP	FIELD2(9),FIELD1
GAMMA1	MVC	40(9,8),30(7)
GAMMA2	MVC	40(NINE,REG8),DEC(7)
GAMMA3	MVC	FIELD2,FIELD1
GAMMA4	MVC	FIELD2(9),FIELD1

ALPHA1, ALPHA2, GAMMA1, and GAMMA2 specify explicit lengths and addresses. ALPHA3 and GAMMA3 specify both implied length and implied addresses. ALPHA4 and GAMMA4 specify explicit length and implied addresses. BETA specifies an explicit length for FIELD2 and an implicit length for FIELD1; both addresses are implied.

Extended Mnemonic Codes

For the convenience of the programmer, the assembly program provides extended mnemonic codes that allow conditional branches to be specified mnemonically as well as through the use of the BC machine instruction. These extended mnemonic codes specify both the machine branch instruction and the condition on which the branch is to occur. The codes are not part of the Universal Set of machine instructions, but are translated by the assembly program into the corresponding operation and condition combinations.

The allowable extended mnemonic codes and their operand formats are shown in the following list, together with their machine instruction equivalents. Unless the instruction ends with an 'R', the extended mnemonics shown are for instructions in the RX format. Note that the only difference between the operand fields of the extended mnemonics and those of their machine instruction equivalents is the absence of the R1 field and the comma that separates it from the rest of the operand field. The extended mnemonic list, like the machine instruction list, shows explicit address formats only. Each address can be specified as an implicit address, as explained previously under "Effective Address — — — Explicit and Implied" and as summarized in Tables 3–1 and 3–2. Examples that illustrate instructions using extended mnemonic codes follow the list of extended mnemonics in Table 3–5.

In the following example, which illustrates the use of extended mnemonics, it is to be assumed that the symbol GO is defined elsewhere in the program.

Name	Operation	Operand
	B	40(3,6)
	B	40(0,6)
	BL	GO(3)
	BL	GO
	BR	4

The first two instructions specify an unconditional branch to an explicit address. The address in the first case is the sum of the contents of base register 6, the contents of index register 3, and the displacement 40; the address in the second instruction is not indexed. The third instruction specifies a branch on low to the address implied by GO as indexed by the contents of index register 3; the fourth instruction does not specify an index register. The last instruction is an unconditional branch to the address contained in register 4.

Table 3–5. Extended Mnemonics

Extended Code		Meaning	Machine Instruction	
B	D2(X2,B2)	Branch Unconditional	BC	15,D2(X2,B2)
BR	R2	Branch Unconditional (RR Format)	BCR	15,R2
NOP	D2(X2,B2)	No Operation	BC	0,D2(X2,B2)
NOPR	R2	No Operation (RR Format)	BCR	0,R2

Table 3–5. Extended Mnemonics (Continued)

Extended Code		Meaning	Machine Instruction	
		Used After Compare Instructions		
BH	D2(X2,B2)	Branch on High	BC	2,D2(X2,B2)
BHR	R2	Branch on High (RR Format)	BCR	2,R2
BHE	D2(X2,B2)	Branch on a High or Equal	BC	10,D2(X2,B2)
BHER	R2	Branch on a High or Equal (RR Format)	BCR	10,R2
BL	D2(X2,B2)	Branch on Low	BC	4,D2(X2,B2)
BLR	R2	Branch on Low (RR Format)	BCR	4,R2
BLE	D2(X2,B2)	Branch on Low or Equal	BC	12,D2(X2,B2)
BLER	R2	Branch on Low or Equal (RR Format)	BCR	12,R2
BE	D2(X2,B2)	Branch on Equal	BC	8,D2(X2,B2)
BER	R2	Branch on Equal (RR Format)	BCR	8,R2
BNH	D2(X2,B2)	Branch on Not High	BC	13,D2(X2,B2)
BNHR	R2	Branch on Not High (RR Format)	BCR	13,R2
BNL	D2(X2,B2)	Branch on Not Low	BC	11,D2(X2,B2)
BNLR	R2	Branch on Not Low (RR Format)	BCR	11,R2
BNE	D2(X2,B2)	Branch on Not Equal	BC	7,D2(X2,B2)
BNER	R2	Branch on Not Equal (RR Format)	BCR	7,R2
		Used After Arithmetic Instructions		
BO	D2(X2,B2)	Branch on Overflow	BC	1,D2(X2,B2)
BOR	R2	Branch on Overflow (RR Format)	BCR	1,R2
BV	D2(X2,B2)	Branch on Overflow	BC	1,D2(X2,B2)
BVR	R2	Branch on Overflow (RR Format)	BCR	1,R2
BP	D2(X2,B2)	Branch on Plus	BC	2,D2(X2,B2)
BPR	R2	Branch on Plus (RR Format)	BCR	2,R2
BM	D2(X2,B2)	Branch on Minus	BC	4,D2(X2,B2)
BMR	R2	Branch on Minus (RR Format)	BCR	4,R2
BNM	D2(X2,B2)	Branch on Not Minus	BC	11,D2(X2,B2)
BNMR	R2	Branch on Not Minus (RR Format)	BCR	11,R2
BNP	D2(X2,B2)	Branch on Not Plus	BC	13,D2(X2,B2)
BNPR	R2	Branch on Not Plus (RR Format)	BCR	13,R2
BNZ	D2(X2,B2)	Branch on Not Zero	BC	7,D2(X2,B2)
BNZR	R2	Branch on Not Zero (RR Format)	BCR	7,R2
BZ	D2(X2,B2)	Branch on Zero	BC	8,D2(X2,B2)
BZR	R2	Branch on Zero (RR Format)	BCR	8,R2
		Used After Test Under Mask Instruction		
BALL	D2(X2,B2)	Branch if all Ones	BC	1,D2(X2,B2)
BALLR	R2	Branch if all Ones (RR Format)	BCR	1,R2
BSOM	D2(X2,B2)	Branch if some Ones	BC	4,D2(X2,B2)
BSOMR	R2	Branch if some Ones (RR Format)	BCR	4,R2
BNON	D2(X2,B2)	Branch if no Ones	BC	8,D2(X2,B2)
BNONR	R2	Branch if no Ones (RR Format)	BCR	8,R2
BNO	D2(X2,B2)	Branch if not all Ones	BC	14,D2(X2,B2)
BNOR	R2	Branch if not all Ones (RR Format)	BCR	14,R2